

US008996747B2

(12) **United States Patent**
Bajpai et al.

(10) **Patent No.:** **US 8,996,747 B2**
(45) **Date of Patent:** **Mar. 31, 2015**

(54) **METHODS AND PHYSICAL COMPUTER-READABLE STORAGE MEDIA FOR INITIATING RE-ENUMERATION OF USB 3.0 COMPATIBLE DEVICES**

(75) Inventors: **Pradeep Bajpai**, San Jose, CA (US);
Robert Rundell, Bellevue, WA (US)

(73) Assignee: **Cypress Semiconductor Corporation**,
San Jose, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

7,500,027	B2	3/2009	Wu	
7,694,032	B2	4/2010	Lim et al.	
7,707,323	B2*	4/2010	Matton et al.	710/8
7,711,863	B2	5/2010	Matton et al.	
7,752,029	B2	7/2010	Tamayo et al.	
7,908,335	B1	3/2011	Citterelle et al.	
8,041,866	B2	10/2011	Barake et al.	
8,171,502	B2	5/2012	Mullis, II et al.	
8,176,227	B2	5/2012	Devam et al.	
8,219,729	B1*	7/2012	Wright	710/104
8,843,664	B2	9/2014	Bajpai et al.	
2006/0064513	A1*	3/2006	Creigh	710/11
2007/0115831	A1	5/2007	Sharma et al.	
2009/0222685	A1	9/2009	Foster et al.	
2009/0271533	A1	10/2009	Asnaashari	

(Continued)

(21) Appl. No.: **13/248,326**

(22) Filed: **Sep. 29, 2011**

(65) **Prior Publication Data**

US 2013/0086282 A1 Apr. 4, 2013

(51) **Int. Cl.**

G06F 13/14 (2006.01)
G06F 11/30 (2006.01)
G06F 13/42 (2006.01)
G06F 9/44 (2006.01)

(52) **U.S. Cl.**

CPC **G06F 11/3041** (2013.01); **G06F 11/3055**
(2013.01); **G06F 13/426** (2013.01); **G06F**
9/4413 (2013.01); **G06F 9/4411** (2013.01)
USPC **710/10**; 710/8

(58) **Field of Classification Search**

CPC **G06F 9/4411**; **G06F 9/4413**
USPC **710/10**
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,647,452 B1 11/2003 Sonoda
6,738,834 B1 5/2004 Williams et al.

FOREIGN PATENT DOCUMENTS

WO 2010132944 A 11/2010

OTHER PUBLICATIONS

Hewlett-Packard et al. Universal Serial Bus 3.0 Specification. Jun. 6, 2011 Revision 1.0. pages 3-6, 6-34, and 7-13. The full document is available on <http://www.usb.org/developers/docs/>.*

(Continued)

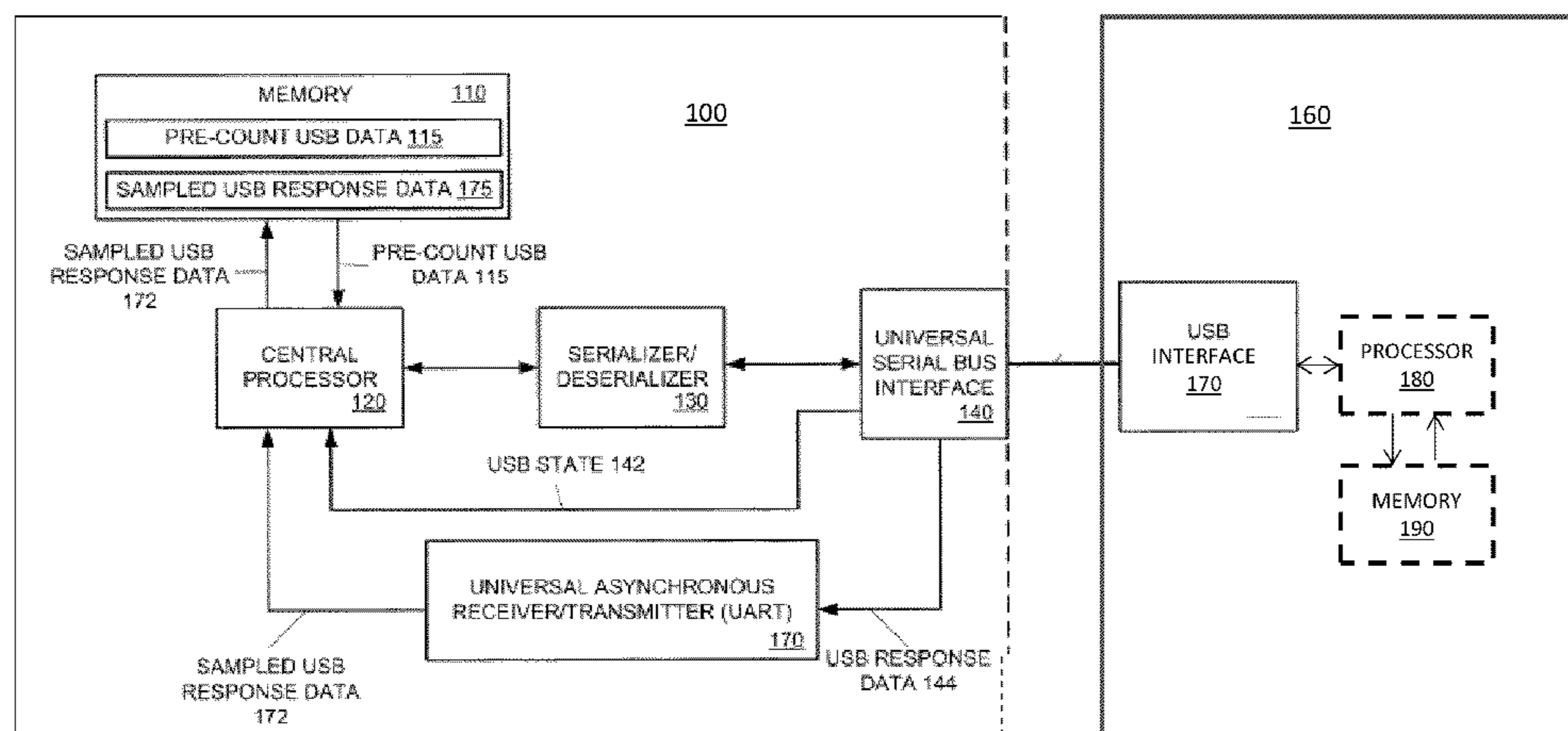
Primary Examiner — Henry Tsai

Assistant Examiner — John Roche

(57) **ABSTRACT**

Methods, physical computer-readable media, and devices are provided that allow re-enumeration to be initiated on a USB 3.0-compatible device. The method includes establishing a connection with a host, transmitting an indicator from the device to the host to cause a Link Training and Status State Machine (LTSSM) of the host to move from active state (U0) to one of SS.Inactive and RX.Detect, synchronizing the device with the host, and presenting a new configuration of the device to the host.

20 Claims, 2 Drawing Sheets



(56)

References Cited

U.S. PATENT DOCUMENTS

2010/0093401 A1 4/2010 Moran et al.
 2010/0122021 A1 5/2010 Lee et al.
 2010/0180051 A1 7/2010 Harris
 2010/0180138 A1 7/2010 Yen et al.
 2010/0185808 A1 7/2010 Yu et al.
 2010/0223402 A1 9/2010 Chen
 2010/0275037 A1 10/2010 Lee et al.
 2010/0312929 A1* 12/2010 Liu et al. 710/63
 2010/0328493 A1 12/2010 Hanlon et al.
 2011/0066266 A1 3/2011 Nakano
 2011/0093633 A1 4/2011 Solomon et al.
 2011/0106989 A1 5/2011 Solomon et al.
 2011/0113079 A1 5/2011 Jou et al.
 2011/0126005 A1 5/2011 Carpenter et al.
 2011/0161530 A1 6/2011 Pietri et al.
 2011/0179201 A1 7/2011 Monks et al.
 2012/0008938 A1* 1/2012 Tang et al. 398/9
 2012/0020404 A1 1/2012 Hsieh et al.
 2012/0023318 A1* 1/2012 Xing et al. 713/2
 2012/0102244 A1* 4/2012 Shiraishi 710/63
 2012/0311122 A1* 12/2012 Johnsen et al. 709/223
 2013/0086282 A1 4/2013 Bajpai et al.

OTHER PUBLICATIONS

Written Opinion of the International Searching Authority for International Application No. PCT/US11/54241 dated Oct. 18, 2012; 5 pages.
 International Search Report for International Application No. PCT/US11/54241 dated Oct. 18, 2012; 3 pages.
 International Search Report for International Application No. PCT/US2012/072204 mailed Mar. 8, 2013; 2 pages.

USB 3.0 Connection Exerciser; online Sep. 2012; <http://www.mcci-jp.com/devtools/exerciser.html>; MCCI product page; 3 pages.
 USB Background—Knowledge Base—Article 10047—Total Phase, Inc., Jun. 17, 2012, pp. 1-16. Retrieved from the internet <URL:<http://web.archive.org/web/20120617132016/http://www.totalphase.com/support/kb/10047/>>.
 USB Device Disconnect-On-Demand with uPSD32xx; online <http://www.icbase.com/pdf/STM/STM30020501.pdf>; STMicroelectronics; copyright 2004; 11 Pages.
 USPTO Advisory Action for U.S. Appl. No. 13/632,084 dated Aug. 2, 2013; 3 pages.
 USPTO Final Rejection for U.S. Appl. No. 13/632,084 dated Jun. 24, 2013; 18 pages.
 USPTO Non-Final Rejection for U.S. Appl. No. 13/632,084 dated Jan. 29, 2013; 16 pages.
 USPTO Non-Final Rejection for U.S. Appl. No. 13/632,084 dated Oct. 24, 2013; 20 pages.
 Written Opinion of the International Searching Authority for International Application No. PCT/US2012/072204 mailed Mar. 8, 2013; 3 pages.
 U.S. Appl. No. 13/632,084: “Re-Enumeration of USB 3.0 Compatible Devices” Pradeep Kumar Bajpai et al., filed Sep. 30, 2012; 49 pages.
 USPTO Final Rejection for U.S. Appl. No. 13/632,084 dated Mar. 24, 2014; 11 pages.
 USPTO Notice of Allowance for U.S. Appl. No. 13/632,084 dated May 15, 2014; 10 pages.
 USPTO Notice of Allowance for U.S. Appl. No. 13/632,084 dated Jun. 24, 2014; 8 pages.
 USPTO Non Final Rejection for U.S. Appl. No. 14/491,809 dated Feb. 2, 2015; 26 pages.

* cited by examiner

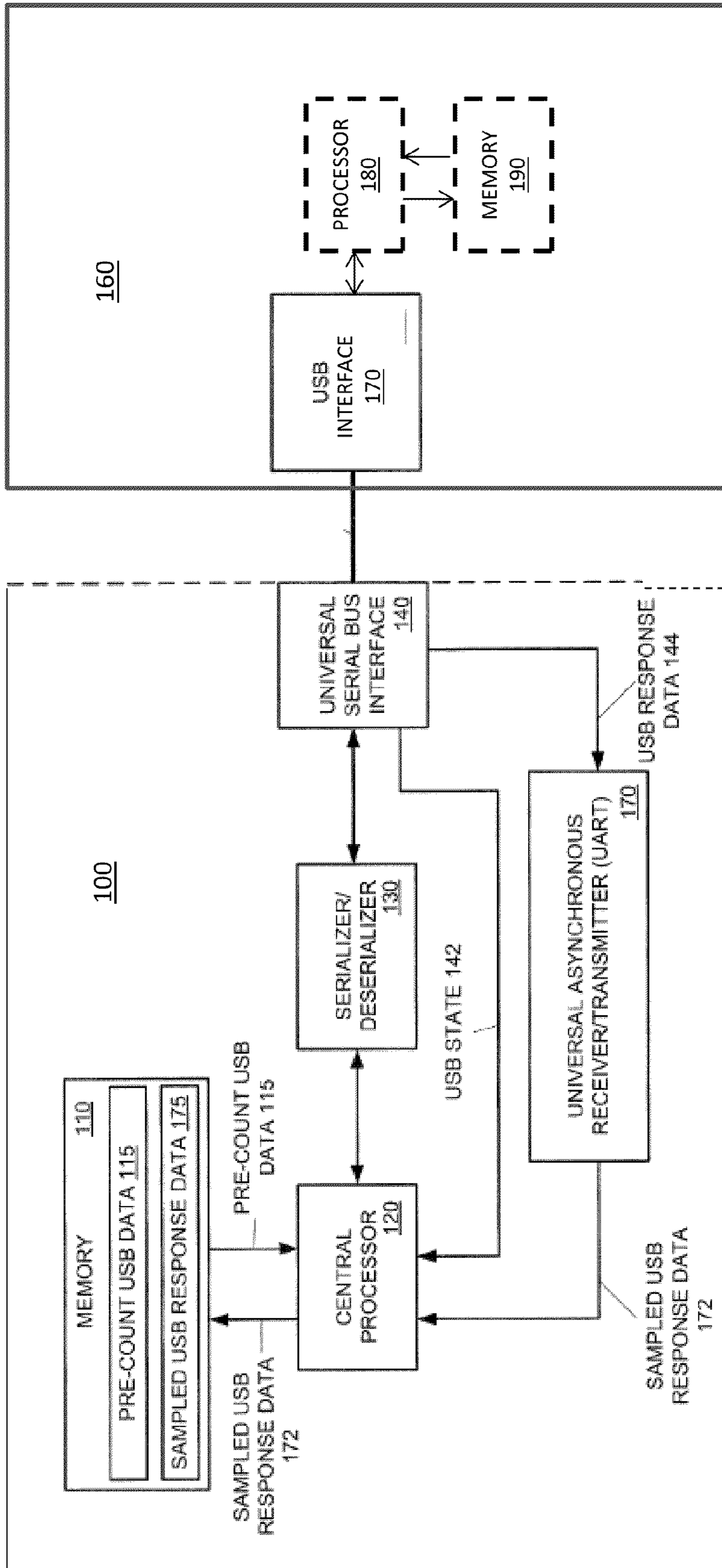


FIG. 1

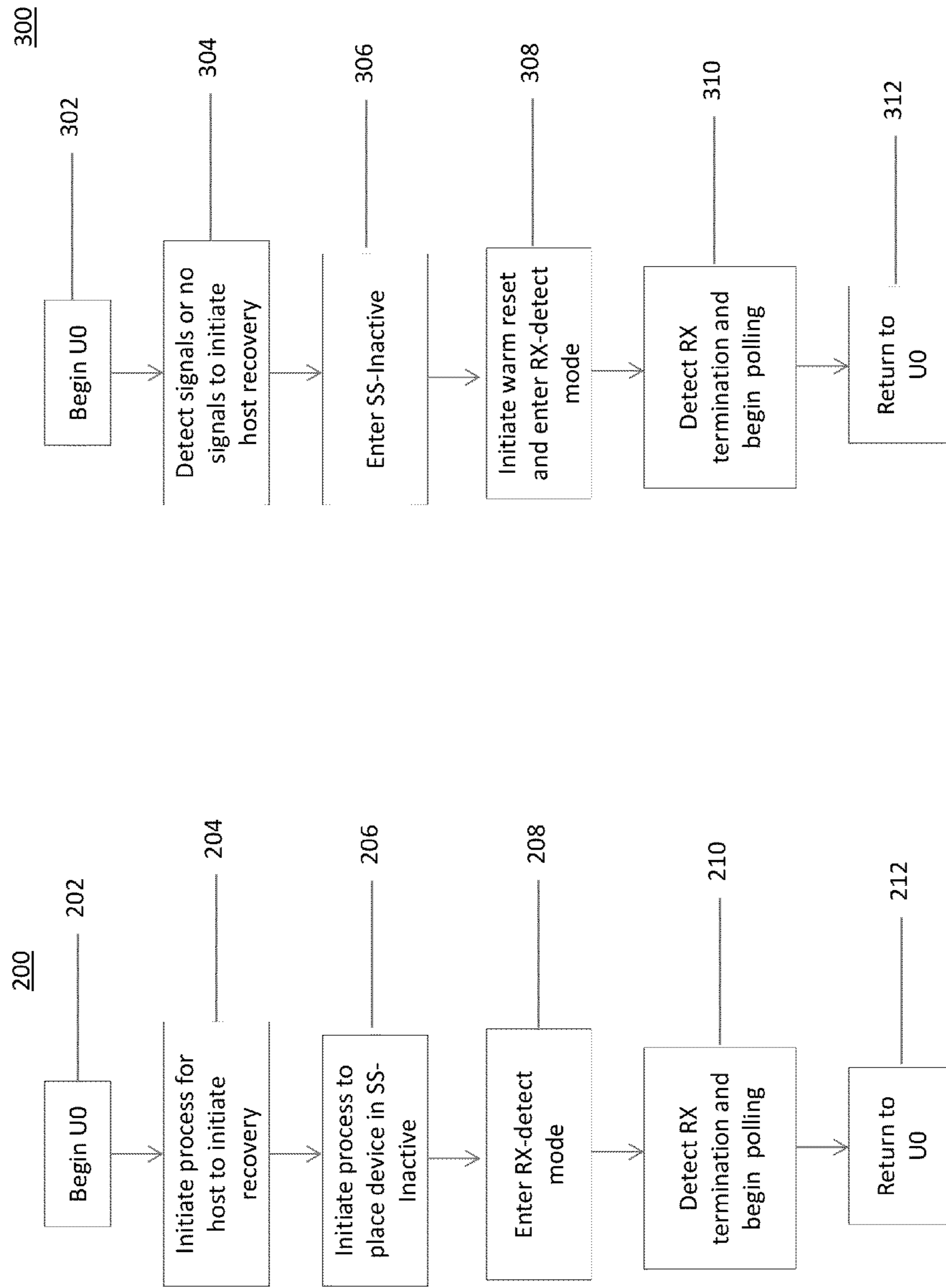


FIG. 2

FIG. 3

1

**METHODS AND PHYSICAL
COMPUTER-READABLE STORAGE MEDIA
FOR INITIATING RE-ENUMERATION OF
USB 3.0 COMPATIBLE DEVICES**

TECHNICAL FIELD

This disclosure relates to the field of universal serial buses (USBs) and, in particular, to re-enumeration of a device compatible with USB 3.0.

BACKGROUND

The Universal Serial Bus (USB) standard provides a universal interface for a Personal Computer (PC) that includes universal plug-and-play and relative ease-of-use. Specifically, when a USB peripheral device is plugged-in to a USB port on a PC, the system will auto-detect and auto-configure the peripheral device. The USB peripheral devices may include devices such as printers, scanners, keyboards, a mouse, joysticks, digital cameras, digital video cameras, data acquisition devices, modems, speakers, telephones or video phones, storage devices such as ZIP drives, or any other peripheral or computing device. In most cases, there is zero user intervention. The USB interface also eliminates the need for multiple input/output (I/O) standards to peripheral devices, thereby simplifying PC connectivity for the consumer as well as simplifying manufacturing for the PC Original Equipment Manufacturers (OEMs).

The original USB specification has evolved over time to meet the needs of industry, resulting in three versions available today. The first two versions, USB 1.0 (later revised to USB 1.1) and USB 2.0, respectively, are wired interfaces, as they use a cable between a host (for example a personal computer or PC) and the USB peripheral device. Although the USB 1.1 has a top transmission speed of 12 million bits/second (Mb/s), system performance may become sluggish if multiple multimedia devices are attached to one USB 1.1 port. USB 2.0 transmission speed is increased to 480 mega bits/second but maintains backwards compatibility to the full speed (12 Mb/s) and low speed (1.5 Mb/s) devices defined in the USB 1.1 specification. USB 3.0 has transmission speeds of up to 5 gigabits/second and has downward compatibility to USB 2.0 devices.

Although USB 3.0 provides many improvements over previous USB versions, facets of its implementation may not be explicitly defined and may be improved and/or extended. For example, at present, there is no specification-compliant method in existence to perform re-enumeration.

BRIEF DESCRIPTION OF THE DRAWINGS

The present disclosure is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings.

FIG. 1 is a block diagram of a universal serial bus (USB) host system connected to a device, according to an embodiment.

FIG. 2 is a flow diagram of a method of inducing a host to perform re-enumeration on a device, according to an embodiment.

FIG. 3 is a flow diagram of a method of performing re-enumeration on a device by a host, according to an embodiment.

DETAILED DESCRIPTION

Reference in the description to “one embodiment” or “an embodiment” means that a particular feature, structure, or

2

characteristic described in connection with the embodiment is included in at least one embodiment of the invention. The phrase “in one embodiment” located in various places in this description does not necessarily refer to the same embodiment.

In the following detailed description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the subject matter of the present application. It will be evident, however, to one skilled in the art that the disclosed embodiments, the claimed subject matter and their equivalents may be practiced without these specific details.

The detailed description includes references to the accompanying drawings, which form a part of the detailed description. The drawings show illustrations in accordance with example embodiments. These embodiments, which may also be referred to herein as “examples,” are described in enough detail to enable those skilled in the art to practice the embodiments of the claimed subject matter described herein. The embodiments may be combined, other embodiments may be utilized, or structural, logical, and electrical changes may be made without departing from the scope and spirit of the claimed subject matter. The following detailed description is not to be taken in a limiting sense as the scope of the subject matter to be patented is defined by the appended claims and their equivalents.

A method is provided that allows re-enumeration to be initiated on a USB 3.0 compatible device. The method includes establishing a connection with a host, transmitting an indicator from the device to the host to cause a Link Training and Status State Machine (LTSSM) of the host to move from active state (U0) to one of SS.Inactive and RX.Detect, synchronizing the device with the host, and presenting a new configuration of the device to the host. By transmitting the indicator from the device to the host, a disconnection between the device and host is emulated. As a result, the host responds by reconnecting to the device and performing re-enumeration on the device. The method may be implemented into a system including at least one host and device.

FIG. 1 is a block diagram of a universal serial bus (USB) host system **100** connected to a device **160**, according to an embodiment. The USB host system **100** is coupled to the device **160** via a bus **150**. The bus **150** may include a USB cable, PCB traces, or other electrical physical connections. In other embodiments, the bus **150** and device **160** are configured to comply with USB 3.0 standards delineated in Universal Serial Bus 3.0 Specification, revision 1.0, Nov. 12, 2008, which may also be found at <http://www.usb.org/developers/docs>.

The USB host system **100** includes a central processor **120** to control the communication with the device **160** for the USB host system **100**. For instance, the central processor **120** may direct the USB host system **100** to issue requests over the bus **150**, synchronize with incoming data from the bus **150** by tracking synchronization bits in the incoming data, decode the incoming data, determine whether the incoming data was received correctly, and respond to the incoming data when necessary. In an embodiment, the central processor **120** is a programmed processor (e.g., a central processing unit), hard-wired, or fixed-function logic. In other embodiments, the central processor **120** has a different configuration.

The USB host system **100** may be a reduced functionality USB host controller, capable of performing a set of one or more preprogrammed functions from the USB specification. For instance, when the USB host system **100** is incorporated into a host device, such as a battery charger, the USB host system **100** may include functionality that allows the host

device to charge the battery of a device **160** (such as a cell-phone, personal digital assistant, digital cameras, digital video cameras, and the like) via the bus **150**. In another example, the USB host system **100** may include functionality that allows the host device to communicate commands to begin operation of a device **160** (such as a printer, a camera, a mouse, printers, scanners, keyboards, joysticks, data acquisition devices, modems, speakers, telephones or video phones, storage devices such as ZIP drives or another peripheral device).

The USB host system **100** includes a memory **110** to store USB data **115** for use in transmissions over the bus **150** to the device **160**. This USB data **115** may be generated and stored in memory **110** by the central processor **120** or by another device internal to or external from the USB host system **100**. The USB data **115** may have a format that allows the USB host system **100** to transfer the USB data **115** from the memory **110** to the bus **150** without having to perform alterations on the USB data **115**.

The USB host system **100** includes a USB physical interface **140** to couple to the bus **150**. The USB interface **140** may identify when a device (e.g., device **160**) is coupled to the USB host system **100** via the bus **150** and provide a USB state **142** to the central processor **120** that indicates the device **160** is coupled to the USB host system **100** via the bus **150**. The bus **150** may have 2 states: a J state and a K state. The USB interface **140** may identify when a device **160** is coupled to the USB host system **100** according to the presence of one of these USB states, or from transitions in the USB states.

The central processor **120** may receive USB data **115** from the memory **110**, e.g., responsive to the USB state **142**, and direct the USB host system **100** to provide the USB data **115** on the bus **150**. In some embodiments, the USB data **115** may be a request for the device **160**, or may be an acknowledgement of USB response data **144** received from the device **160**. The USB data **115** may be a complete bit sequence or substantially complete bit sequence that is available for transmission over the bus **150**. For instance, when the USB data **115** is a complete bit sequence, the USB host system **100** may directly retrieve the USB data **115** from memory **110** and send it over the bus **150** without having to perform additional processing on the data. When the USB data **115** is a substantially complete bit sequence, the USB host system **100** may directly retrieve the USB data **115** from memory **110** and send it over the bus **150** with little additional processing, such as appending a preamble, etc.

The USB host system **100** includes a serializer/deserializer **130** to perform serialization operations on outgoing data and deserialization operations on data incoming from the bus **150**. The USB host system **100** may also include a universal asynchronous receiver and transmitter (UART) **170** to sample USB response data **144** from the device **160**. In some embodiments, the central processor **120** or other device in the USB host system **100** may sample USB response data **144** from the device **160**. The UART **170** may over-sample the USB response data **144**, for example using a 4 times over-sampling process, to recover the response and to generate sampled USB response data **172**. The UART **170** may provide the sampled USB response data **172** to the central processor **120** via the serializer/deserializer **130** for storage and processing.

When the central processor **120** receives the USB state **142** indicating that the USB peripheral device **160** is coupled to the USB host system **100**, the central processor **120** may retrieve a pre-generated Get_Device_Descriptor request, which is a standard USB request, from the memory **110**. Once the pre-generated Get_Device_Descriptor request is provided to the device **160** over the bus **150**, the device **160** may

generate a response to the pre-generated Get_Device_Descriptor request. The response may include a device descriptor that describes the type of device coupled to the USB host system **100** via the bus **150**.

The USB host system **100** may receive the response at the USB interface **140** as USB response data **144** and provide the USB response data **144** to the UART **170**. The UART **170** may sample the USB response data **144** and provide the sampled USB response data **172** to the central processor **120** via the serializer/deserializer **130** for storage and/or processing. In some embodiments, the UART **170** may over-sample the USB response data **144**, for example, using a 4 times over-sampling process, to generate sampled USB response data **172**.

The central processor **120** may direct the USB host system **100** to provide an acknowledgement to the response from the device **160**. The acknowledgement may be stored in the memory **110** as pre-generated USB data **115**, which is retrieved responsive to the reception of the response from the device **160**. In some embodiments, the acknowledgement is sent to the device **160** over the bus **150** prior to the central processor **120** parsing the response to identify the device descriptor.

The central processor **120** may store the sampled USB response data **172** in the memory **110**, and then subsequently retrieve the sampled USB response data **172** for processing. The central processor **120** may parse the sampled USB response data **172** to determine whether the USB response data **144** was correctly received by the USB host system **100**. For instance, the central processor **120** may perform a cyclical redundancy check (CRC) and compare the results of the CRC to the contents of a CRC field in the USB response data **144**.

As delineated in the USB 3.0 protocol, a USB 3.0 compatible USB host system **100** is configured to detect the device **160** at least twice when connected. Repetitive detection allows the USB host system **100** to perform re-enumeration on the device **160** to thereby obtain any updated device configuration data.

The device **160** may comprise a USB peripheral device and include a USB interface **170**. The USB interface **170** may indicate when the device **160** is coupled to the USB host system **100** via the bus **150** and provide a USB state that indicates the device **160** is coupled to the USB host system **100** via the bus **150**.

Depending on the particular configuration and purpose, the device **160** may include various components for carrying out the operation of the device **160**. For example, the device may be an input device, such as a keyboard, pointing device (e.g., mouse, light pen, joystick, touch screen, gaming devices, and the like), imaging device (e.g., a webcam, videocam, scanner, and the like), or an audio device (e.g., microphone) for providing data and control signals to a host system **100** via a device processor **180**. Thus, the input device may include buttons, cameras, optical readers, or other components in communication with the device processor **180**.

In another example, the device **160** may be an output device that delivers an output to a user resulting from processing completed by the host system **100**. For example, the output device may include speakers, headphones, video screen, and the like and may include operational components associated with such devices. Those operational components may be in communication with the device processor **180**, in an embodiment.

In other embodiments, the device **160** includes a memory **190**, which may include software stored therein for processes

5

that present device configuration data to the USB host system 100 to thereby allow the USB host system 100 to identify the device 160.

Although the processor 180 and memory 190 are depicted in phantom, it will be appreciated that one or both components may be included in the device 160. In other embodiments, additional components, including but not limited to those described above, may also be included as part of the device 160.

When a host (e.g., host system 100) is connected to a device (e.g., device 160), the host reads device descriptors to determine the capabilities of the device to thereby allow drivers to be loaded onto the device. In this regard, the host performs an enumeration protocol on the device. If the configuration descriptors of the device need to be re-read, the host performs re-enumeration on the device. To induce the host into performing re-enumeration on the device, the device transmits an indicator to the host to cause a host Link Training and Status State Machine (LTSSM) to move from active state (U0) to one of SS.Inactive and RX.Detect. In an embodiment, the host LTSSM moves to RX.Detect after SS.Inactive. In any case, after RX.Detect, the host LTSSM continues to polling and then to U0.

Several methods may be used to induce the host to perform re-enumeration. FIG. 2 is a flow diagram of a method 200 of inducing a host to perform re-enumeration on a device, according to an embodiment. The device begins a U0 state, step 202. For example, the device (e.g., device 160) enters a U0 link state, which indicates that the device is active. In the U0 state, a connection between the device and host (e.g., host system 100) may be established as a default configuration.

Next, the device directs initiation of a process for the host to initiate recovery, step 204. For example, the device may provide error messages to the host, such as by stopping LUP transmission if the USB (e.g., bus 150) is IDLE. In another embodiment, the device may provide error messages that stop LGOOD and LCRD transmissions to cause timeout at a link layer of the host or that include a corrupt sequence number on LGOOD, LCRD, or on a header packet. In still yet another embodiment, the instructions may cause the device to engage in a successful LFPS exit handshake from a low power state (e.g., U1, U2, or U3) to recovery or during the handshake may respond with an improper LFPS. In still yet another embodiment, the device may be instructed to initiate recovery without detecting any error.

In any case, the firmware directs initiation of a process to place the device into an SS.Inactive mode, step 206. As a result, the device receives a signal indicating a warm reset has been initiated by the host. After the warm reset signal is received, the device enters an RX.Detect mode, step 208. At this point, the device resets. Additionally, the device and the host, which is also in an RX.Detect mode, synchronize. After the device and host are synchronized, the device detects RX termination from the host and begins polling, step 210. Training completes after polling and the device returns to a U0 state, step 212. At the U0 state, the device and the host re-establish a connection, and the device presents a new configuration to the device. Specifically, the device (e.g., device 160) may receive a pre-generated Get_Device_Descriptor request from the host 100 over the bus 150 and may generate a response to the pre-generated Get_Device_Descriptor request. The response may include a device descriptor that describes the type of device coupled to the host 100 via the bus 150.

FIG. 3 is a flow diagram of a method 300 of performing re-enumeration on a device by a host, according to an embodiment. Some of the steps of method 300 may occur substan-

6

tially simultaneously with those of method 200. The host begins a U0 state, step 302. When the host is in U0 state, it may be connected with the device. For example, a bus (e.g., bus 150) connects a USB physical interface (e.g., USB physical interface 140) of the host (e.g., host system 100) to a USB physical interface (e.g., USB physical interface 170) of the device (e.g., device 160). Next, the host receives an indication from the device, which causes the host to initiate recovery, step 304. Specifically, the indications include, but are not limited to those described above in conjunction with step 204 of method 200 in FIG. 2. As noted in the above description, in some embodiments, the host enters a low power state (e.g., U1, U2, or U3) after U0 and prior to entry into recovery. After recovery, the host enters an SS.Inactive mode, step 306. In an embodiment, the host enters a hot reset prior to moving to SS.Inactive. In another embodiment, the host enters loopback prior to moving to SS.Inactive. Subsequently, the host initiates a warm reset and enters an RX.Detect mode, step 308.

In other embodiments of the method 300, step 304 is omitted. For example, the host Link Training and Status State Machine (LTSSM) may respond to the device by moving from U0 to U1, U2 or U3 to SS.Inactive mode. In another embodiment, step 306 is omitted and the host moves from U0 to recovery to loopback to RX.Detect mode. In any case, during the RX.Detect mode, the host enters warm reset, and the host and the device synchronize.

After the host and device are synchronized, the host detects RX termination from the device and begins polling, step 310. Training completes after polling and the host returns to a U0 state, step 312. Subsequently, the host receives a new configuration from the device to begin re-enumeration on the device. For example, a central processor (e.g., central processor 120 of the host 100 in FIG. 1) may retrieve a pre-generated Get_Device_Descriptor request, which is a standard USB request, from a memory (e.g., memory 110 of FIG. 1) causing the host to retrieve the new device configurations.

By manipulating link connectivity and link power management (e.g., via the link training and status state machine (LTSSM)), the device emulates disconnect and/or connect between the device and host. As a result, a USB 3.0 host and USB 3.0 device may reconnect with each other and the device may be recognized as a new device. Accordingly, re-enumeration of the device may be performed by the host.

What is claimed is:

1. A method of initiating re-enumeration of a device compatible with a Universal Serial Bus (USB) 3.0 specification, the method comprising:
 - establishing, by the device, a connection with a host over a USB;
 - determining, by the device, to induce re-enumeration of the device by the host over the USB in response to the device having a new configuration to present the host;
 - transmitting, by the device, an indicator to the host over the USB, the indicator being a USB 3.0 indicator emulating to the host a condition of the device that is not a current condition of the device, wherein the transmitting of the emulated condition identified by the indicator induces the re-enumeration of the device by the host over the USB and causes a Link Training and Status State Machine (LTSSM) of the host to move from active state (U0) to one of SS.Inactive and RX.Detect in response to receiving the emulated condition identified by the USB 3.0 indicator;
 - synchronizing, by the device, with the host over the USB;
 - and
 - presenting, by the device, the new configuration of the device to the host over the USB.

7

2. The method of claim 1, wherein transmitting the indicator comprises transmitting a corrupt sequence number.

3. The method of claim 1, wherein the device and the host are connected by the USB, and wherein transmitting the indicator comprises determining whether the USB is idle and preventing, by the device, upstream link command (LUP) transmission, if the USB is idle.

4. The method of claim 1, wherein transmitting the indicator comprises preventing transmission of one or more link commands to the host.

5. The method of claim 4, wherein the one or more link commands include LGOOD and LCRD.

6. The method of claim 1, wherein transmitting the indicator comprises transmitting a signal to the host indicating successful LFPS exit handshake from lower power state U1/U2/U3 to recovery.

7. The method of claim 1, wherein transmitting the indicator comprises sending a signal to the host indicating initiation of recovery without detecting error.

8. An apparatus comprising a non-transitory computer-readable memory device storing instructions configured to cause a controller to perform operations comprising:

establishing a connection with a host over a Universal Serial Bus (USB);

determining to induce re-enumeration of a device including the controller by the host over the USB in response to the device having a new configuration to present the host;

prompt transmission of an indicator to the host over the USB, the indicator being a USB 3.0 indicator emulating to the host a condition of the device that is not a current condition of the device, wherein the transmission of the emulated condition identified by the indicator induces the re-enumeration of the device by the host over the USB and causes a Link Training and Status State Machine (LTSSM) of the host to move from active state (U0) to one of SS.Inactive and RX.Detect in response to receiving the emulated condition identified by the USB 3.0 indicator;

synchronizing the device with the host over the USB; and presenting the new configuration of the device to the host over the USB.

9. The apparatus of claim 8, wherein:

the device and the host are connected by the USB; and the instructions are configured to cause the controller to perform operations further comprising determining whether the USB is idle and preventing upstream link command (LUP) transmission, if the USB is idle.

10. The apparatus of claim 8, wherein the instructions are configured to cause the controller to perform operations further comprising preventing transmission of one or more link commands to the host.

11. The apparatus of claim 8, wherein the instructions are configured to cause the controller to perform operations fur-

8

ther comprising transmitting a signal to the host indicating successful LFPS exit handshake from lower power state U1/U2/U3 to recovery.

12. The apparatus of claim 8, wherein the instructions are configured to cause the controller to perform operations further comprising sending a signal to the host indicating initiation of recovery without detecting error.

13. A device comprising:

a controller; and

a memory in communication with the controller, including instructions, that when executed by the controller, cause the controller to:

establish a connection between the device and a host over a Universal Serial Bus (USB);

determine to induce re-enumeration of the device by the host over the USB in response to the device having a new configuration to present the host;

prompt transmission of an indicator from the device to the host over the USB, the indicator being a USB 3.0 indicator emulating to the host a condition of the device that is not a current condition of the device, wherein the transmission of the emulated condition identified by the indicator induces the re-enumeration of the device by the host over the USB and causes a Link Training and Status State Machine (LTSSM) of the host to move from active state (U0) to one of SS.Inactive and RX.Detect in response to receiving the emulated condition identified by the USB 3.0 indicator;

synchronize the device with the host over the USB; and present the new configuration of the device to the host over the USB.

14. The device of claim 13, wherein the indicator comprises a corrupt sequence number.

15. The device of claim 13, wherein the controller is configured to determine whether the USB is configured to connect the device to the host is idle and prevent upstream link command (LUP) transmission, if the USB is idle.

16. The device of claim 13, wherein the controller is configured to prevent transmission of one or more link commands to the host.

17. The device of claim 16, wherein the one or more link commands include LGOOD and LCRD.

18. The device of claim 13, wherein the controller is configured to transmit a signal to the host indicating successful LFPS exit handshake from lower power state U1/U2/U3 to recovery.

19. The device of claim 13, wherein the controller is configured to send a signal to the host indicating initiation of recovery without detecting error.

20. The device of claim 13, wherein the device comprises a USB 3.0 device.

* * * * *